

## Fișa 1 Subprograme

```
#include <iostream>
#include<math.h>

using namespace std;

int prim1 (int n)
{
    int d,ok=1;
    //ok=1 presupun ca numarul e prim
    //daca n are cel putin un alt divizor decat 1 si
    el insusi ok=0
    for(d=2;d<=sqrt(n);d++)
        if(n%d==0) ok=0;
    return ok;
}

int prim2 (int k)
{
    int d;
    //ok=1 presupun ca numarul e prim
    //daca n are cel putin un alt divizor decat 1 si
    el insusi ok=0
    for(d=2;d<=sqrt(k);d++)
        if(k%d==0) return 0;
    return 1;
}

int main()
{ int n;
  cin>>n;
  return 0;
}
```

- Subprogramele **au un algoritm propriu** pentru rezolvarea unei anumite cerințe? ;
- Ce tip de subprograme sunt? Care este numărul de valori returnate de subprogram?
- Ce returnează ?
- Care este diferența dintre prim1 și prim2?
- Cum ar trebui să comunice modulul apelant cu subprogramul ?

### Sarcini de lucru :

1. Copiați codul sursă alăturat
  2. Compilați
  3. Run
  4. Ce observați ?
  5. În cadrul funcției main(), inserați o valoare.  
Ex. : 4; compilați... 0 erori 1 atenționare (**warning: statement has no effect**)
- ```
int main()
{ int n;
  cin>>n;
  4:
  return 0;
}
```
6. Modificați funcția main() și rulați din nou
- ```
int main()
{ int n;
  cin>>n;
  prim1(n); // ce efect are acest apel ?
  return 0;
}
```
7. Modificați și rulați
- ```
int main()
{ int n,x;
  cin>>n;
  //apel functie operand prin afișare
  cout<<" functia returneaza "<<prim1(n)<<endl;
  //apel functie operand prin operatie de decizie
  if (prim2(n)) cout<<"numarul "<<n<<" este prim";
  else cout<<"numarul "<<n<<" nu este prim";
  cout<<endl;
  // apel functie operand - atribuire
  x=prim1(n);
  cout<<" x= " <<x;
  return 0;
}
```

```

#include <iostream>
#include<math.h>

using namespace std;
int ok;

void prim (int gicu, int &ceva)
{   int d;
    ceva=1;
    for(d=2;d<=sqrt(gicu);d++)
        if(gicu%d==0) ceva=0;
}

int main()
{ int n;
  cin>>n;
  prim(n,ok);
  if (ok) cout<<"numarul "<<n<<" este prim";
    else cout<<"numarul "<<n<<" nu este
    prim";
  return 0;
}

```

// atentie la numele și tipul parametrilor *formali*

- analizați subprogramul `prim`
- cum se apelează?
- Cum se face transferul/comunicarea parametrilor? Ce valoare va avea parametrul `gicu` la intrarea în subprogram?
- Care sunt zonele de memorie alocate ? Ce se reține pe stivă pentru fiecare?
- Se poate renunța la parametrul formal `ceva`? Este corect din punct de vedere al modularizării programelor? Justificați răspunsul.